Explainable AI

How Software and Data Teams Can Get Started

The WHY

Prevent
Black Boxes

Image source. Generated with Copilot in Bing. “A photo realistic image of a black box with elements of magic in the background”

Black Box Example

Saying NO

Photo by Polina Tankilevitch: https://www.pexels.com/photo/colorful-candy-5469042/

Black Box Example - Amazon Resume Screening Tool

“Insight - Amazon scraps secret AI recruiting tool that showed bias against women”

Resumes containing this info were downgraded:

- women ("women's chess club captain.")
- graduation from an all-women’s college

Source: Reuters

Black Box Example - Dogs vs Wolves Image Classification

A husky (on the left) is confused with a wolf, because the pixels (on the right) characterizing wolves are those of the snowy background. This artifact is due to a learning base that was insufficiently representative.

Source: ResearchGate

Black Box Example - Airline Chatbot

Gave false information about discounts for last-minute travel for funerals, causing a larger-than-expected expense for a grieving gentleman.

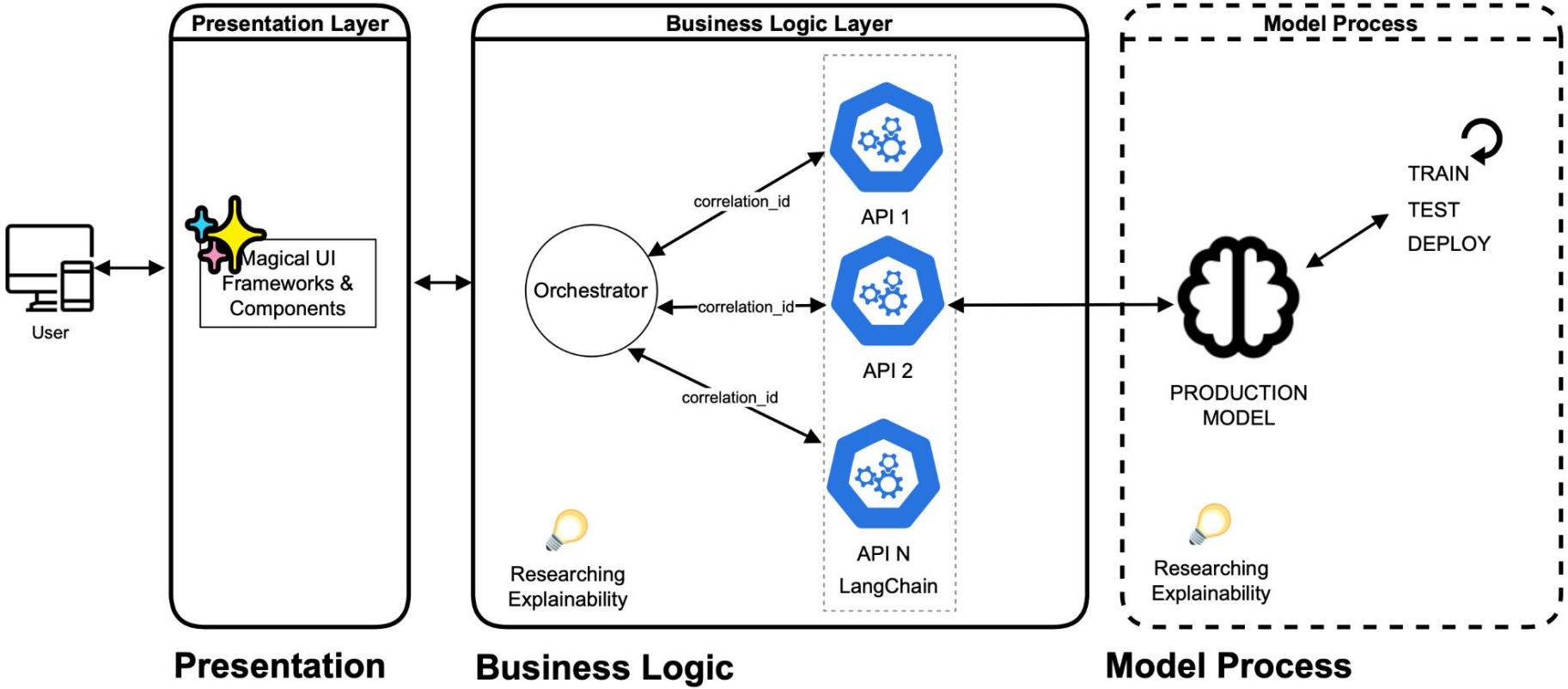
Your Project

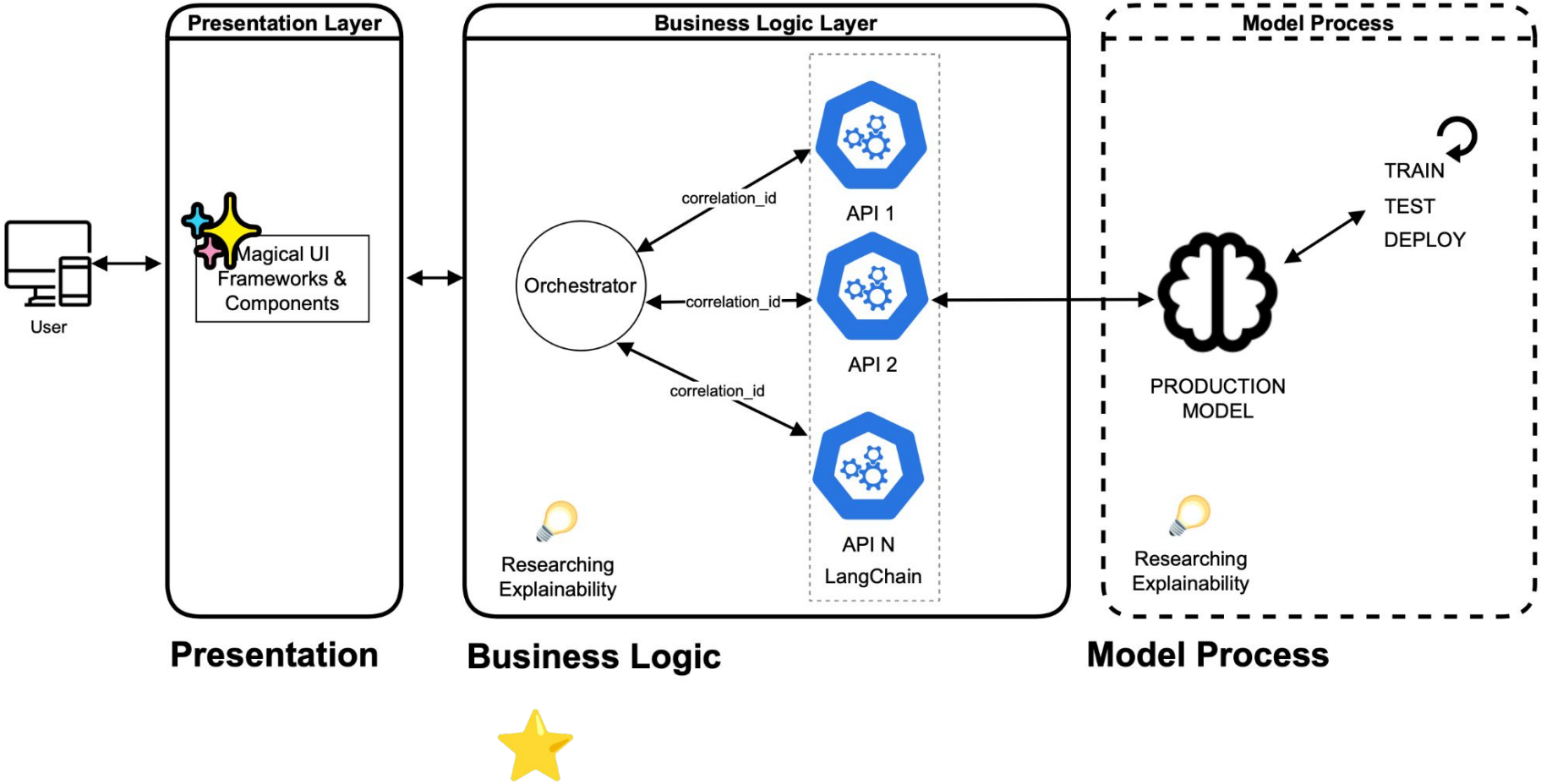
Is Like a Snowflake

Photo by Jakub Sisulak: https://www.pexels.com/photo/white-snow-falling-on-a-person-s-hand-6965965/

There is a business case for knowing how your products work.

Hypothetical Application Architecture




Presentation Layer
Business Logic Layer
Model Process
User
Magical UI
Frameworks &
Components
Orchestrator
correlation_id
correlation_id
correlation_id
API 1
API 2
API N
LangChain
Researching
Explainability
TRAIN
TEST
DEPLOY
PRODUCTION
MODEL
Researching
Explainability
Presentation
Business Logic
Model Process

If you are selecting AI

Recommendations

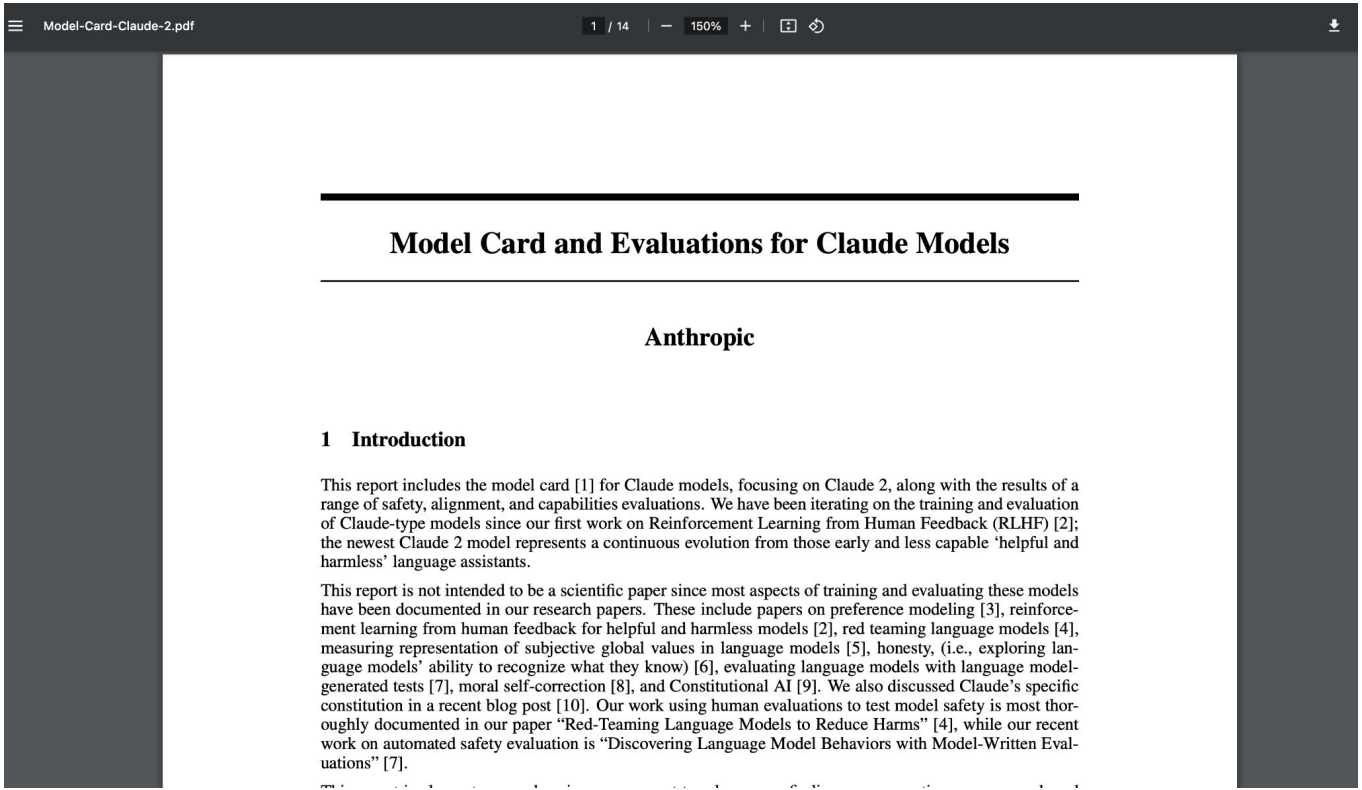
1. Review the model's **Model Card**
2. Log your requests and the responses

Model Cards

What are they?

1. Generic term
2. Document with details about a model
3. May include:
    a. Text description
    b. Intended use
    c. Metrics
    d. Evaluation Data
    e. Ethical concerns
4. Explains the model

Model Card - Claude 2



Model Card and Evaluations for Claude Models

Anthropic

1 Introduction

This report includes the model card [1] for Claude models, focusing on Claude 2, along with the results of a range of safety, alignment, and capabilities evaluations. We have been iterating on the training and evaluation of Claude-type models since our first work on Reinforcement Learning from Human Feedback (RLHF) [2]; the newest Claude 2 model represents a continuous evolution from those early and less capable 'helpful and harmless' language assistants.

This report is not intended to be a scientific paper since most aspects of training and evaluating these models have been documented in our research papers. These include papers on preference modeling [3], reinforcement learning from human feedback for helpful and harmless models [2], red teaming language models [4], measuring representation of subjective global values in language models [5], honesty, (i.e., exploring language models' ability to recognize what they know) [6], evaluating language models with language model-generated tests [7], moral self-correction [8], and Constitutional AI [9]. We also discussed Claude's specific constitution in a recent blog post [10]. Our work using human evaluations to test model safety is most thoroughly documented in our paper "Red-Teaming Language Models to Reduce Harms" [4], while our recent work on automated safety evaluation is "Discovering Language Model Behaviors with Model-Written Evaluations" [7].

Model Card - Claude 2

process. [2, 4, 9]

For this report, we collected data on some common tasks: detailed instruction-following (**helpfulness**); providing accurate, and factual information (**honesty**). We also included a red-teaming task (**harmlessness**), which asked crowdworkers to roleplay adversarial scenarios and trick AI systems into generating harmful content. This approach has its limitations–for instance, we know the scenarios created by crowdworkers are not fully representative of the scenarios Claude will encounter out in real-world usage–but we still consider it a useful data point.

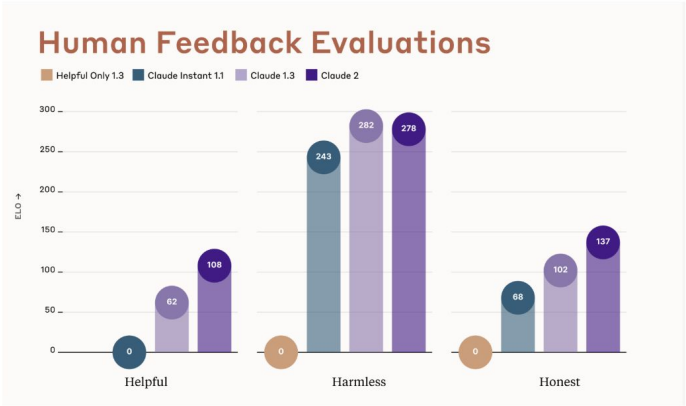


Figure 1 This plot shows task-specific Elo scores, indexed to a baseline model.

Model Cards

How do they help

1. Build trust
2. Compare your data with the training data
3. Does your intended use case match the model's?

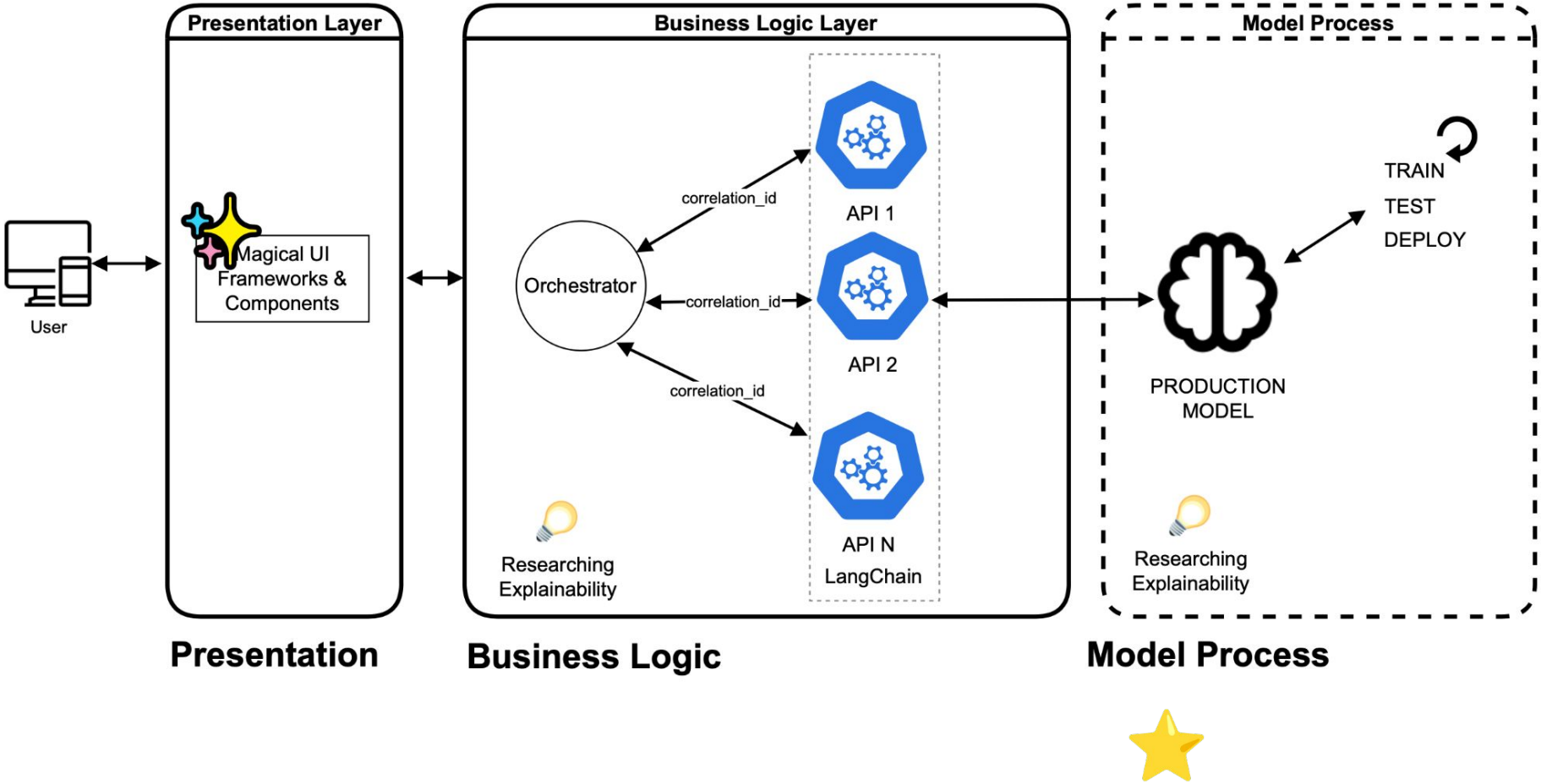
-> Team conversations

Logging

Requests and Responses

1. Need to evaluate the responses to make sure they reasonable.
2. Plan the evaluation as an ongoing maintenance task
3. Make adjustments as needed

Questions?


Presentation Layer
Business Logic Layer
Model Process
User
Magical UI
Frameworks &
Components
Orchestrator
correlation_id
correlation_id
correlation_id
API 1
API 2
API N
LangChain
Researching
Explainability
TRAIN
TEST
DEPLOY
PRODUCTION
MODEL
Researching
Explainability
Presentation
Business Logic
Model Process

If you are developing a model

Actions to Take

1. Data Assessment
2. Interpretability Tooling
3. Model Card

Types of data bias

Types of data bias

From sources across the web

Selection bias	Confirmation bias	Association Bias
Measurement bias	Observer bias	Recall bias
Racial bias	Survivorship bias	Data and information visualization
Evaluation bias	Omitted variable bias	Attrition bias
Publication bias	Overgeneralization bias	Acquiescence bias
Availability heuristic	Algorithmic bias	Performance bias
Self-selection bias	Question order bias	Automation bias
Historic bias	Interviewer bias	Outlier bias

Show less

Feedback

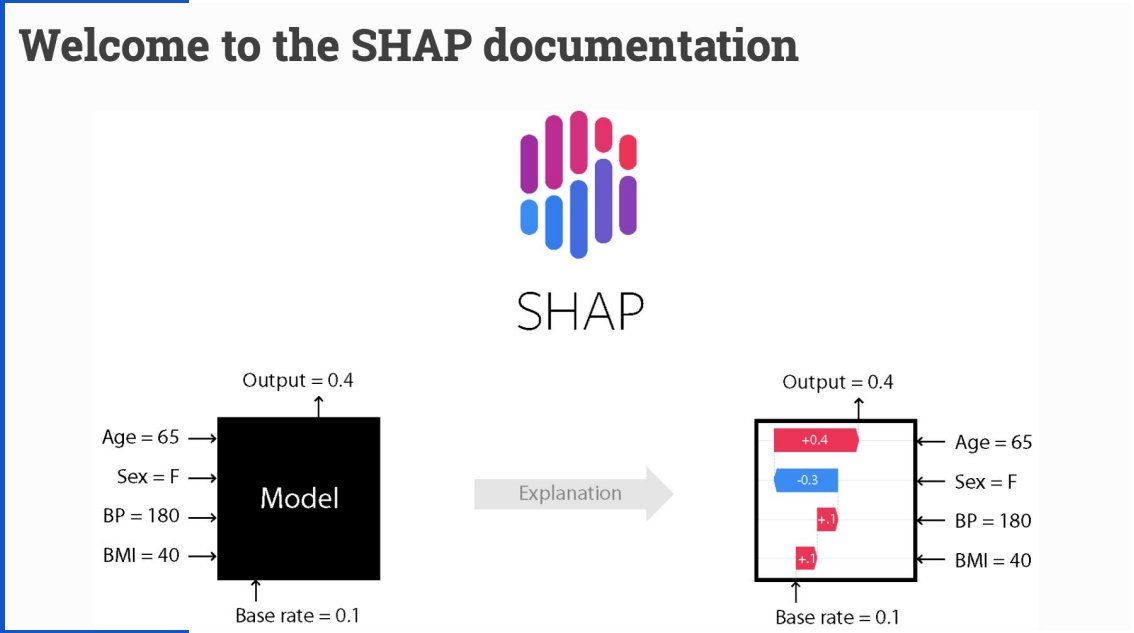
Interpretability

Several approaches

1. SHAP
2. LIME
3. Permutation Importance
4. Partial Dependency Plot
5. Anchor

SHAP

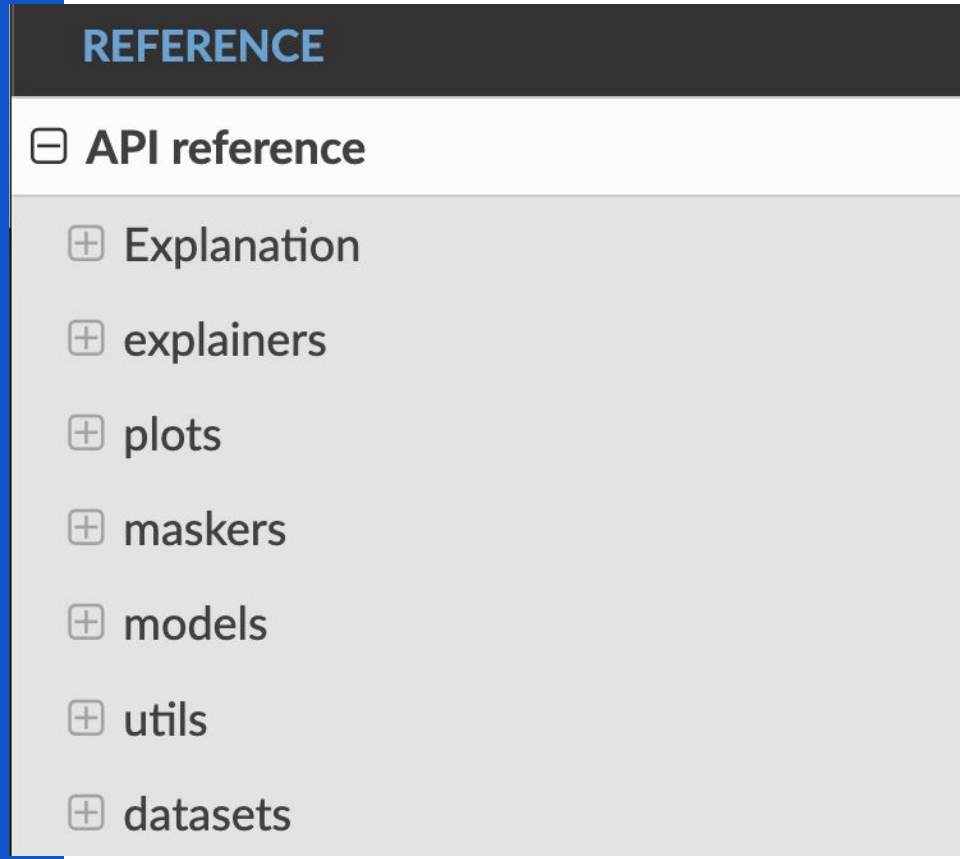
SHapley Additive exPlanations)



Source: https://shap.readthedocs.io/en/latest/index.html

SHAP

API



Source:
https://shap.readthedocs.io/en/latest/api.html

Dataset

shap.datasets.california

shap.datasets.california(*n_points=None*)

Return the California housing data in a structured format.

Parameters: **n_points : int, optional**

Number of data points to sample. If provided, randomly samples the specified number of points.

Returns: **Tuple of pandas DataFrame containing the data and a numpy array representing the target.**

The data include the following features:

- `MedInc` : Median income in block
- `HouseAge` : Median house age in block
- `AveRooms` : Average rooms in dwelling
- `AveBedrms` : Average bedrooms in dwelling
- `Population` : Block population
- `AveOccup` : Average house occupancy
- `Latitude` : House block latitude
- `Longitude` : House block longitude

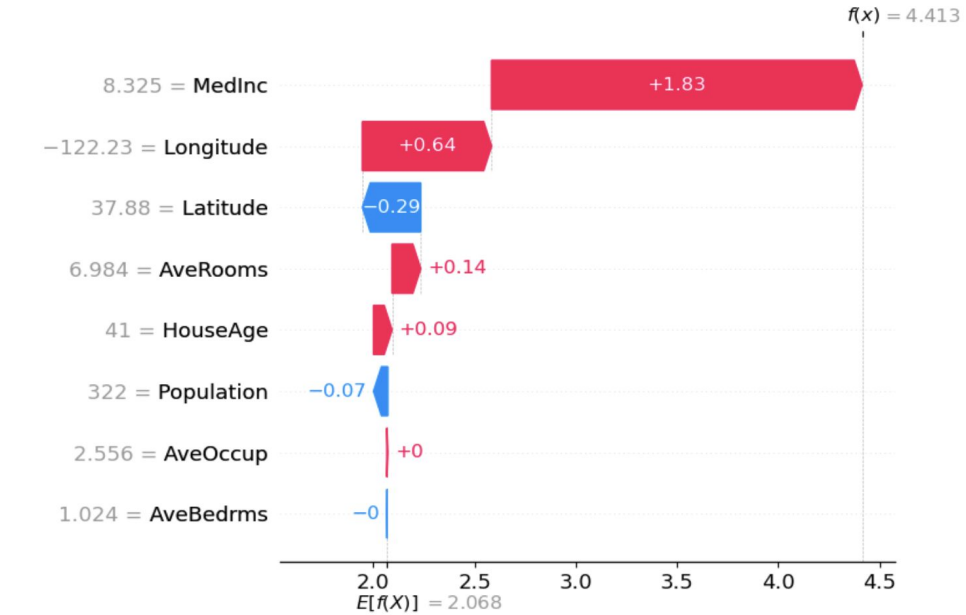
Tree ensemble example (XGBoost/LightGBM/CatBoost/scikit-learn/pyspark models)

While SHAP can explain the output of any machine learning model, we have developed a high-speed exact algorithm for tree ensemble methods (see our Nature MI paper). Fast C++ implementations are supported for *XGBoost*, *LightGBM*, *CatBoost*, *scikit-learn* and *pyspark* tree models:

```
import xgboost
import shap

# train an XGBoost model
X, y = shap.datasets.california()
model = xgboost.XGBRegressor().fit(X, y)

# explain the model's predictions using SHAP
# (same syntax works for LightGBM, CatBoost, scikit-learn, transformers, Spark, etc.)
explainer = shap.Explainer(model)
shap_values = explainer(X)

# visualize the first prediction's explanation
shap.plots.waterfall(shap_values[0])
```

Waterfall Plot - Single Prediction

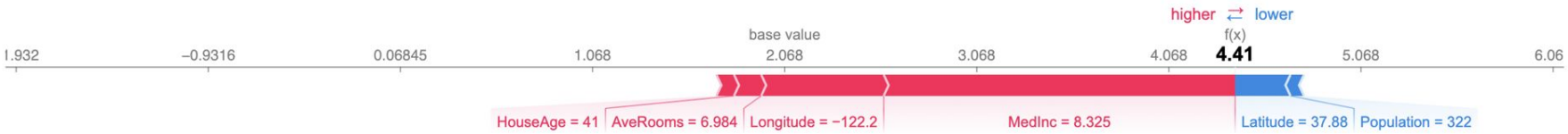
```python
import xgboost
import shap

# train an XGBoost model
X, y = shap.datasets.california()
model = xgboost.XGBRegressor().fit(X, y)

# explain the model's predictions using SHAP
# (same syntax works for LightGBM, CatBoost, scikit-learn, transformers, Spark, etc.)
explainer = shap.Explainer(model)
shap_values = explainer(X)

# visualize the first prediction's explanation
shap.plots.waterfall(shap_values[0])
```



Force Plot - Single Prediction

The above explanation shows features each contributing to push the model output from the base value (the average model output over the training dataset we passed) to the model output. Features pushing the prediction higher are shown in red, those pushing the prediction lower are in blue. Another way to visualize the same explanation is to use a force plot (these are introduced in our Nature BME paper):

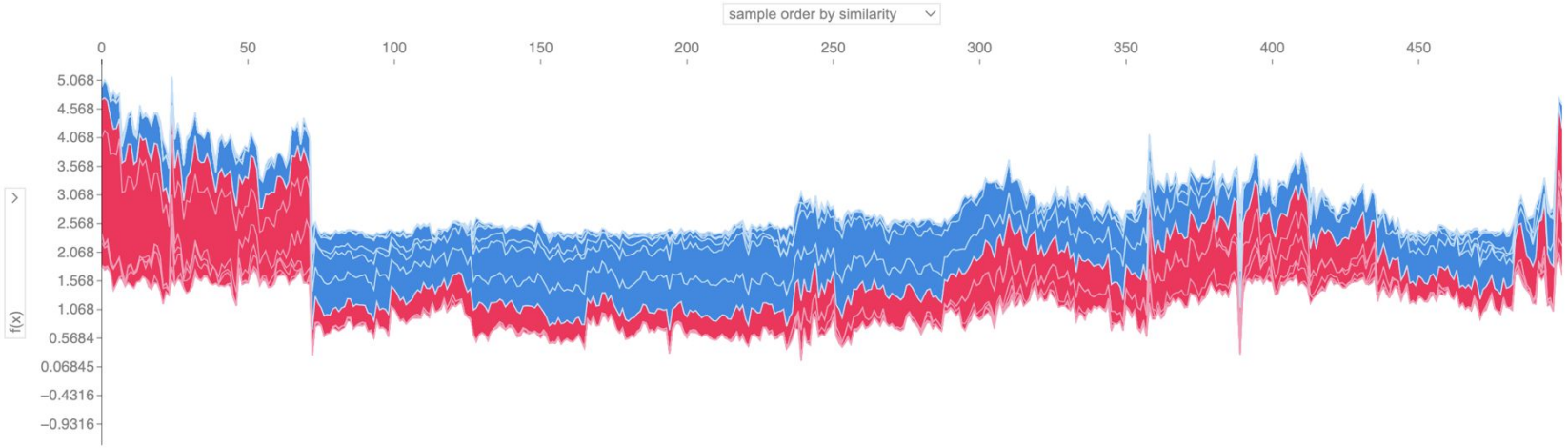
```
# visualize the first prediction's explanation with a force plot
shap.plots.force(shap_values[0])
```



Force Plot - Many Predictions

If we take many force plot explanations such as the one shown above, rotate them 90 degrees, and then stack them horizontally, we can see explanations for an entire dataset (in the notebook this plot is interactive):

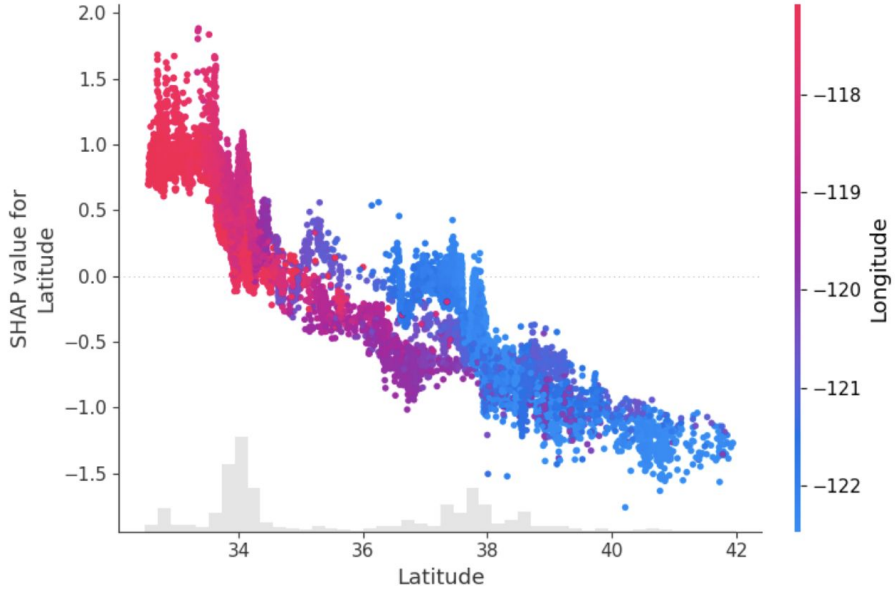
```
# visualize all the training set predictions
shap.plots.force(shap_values[:500])
```



Scatter Plot - Single Feature

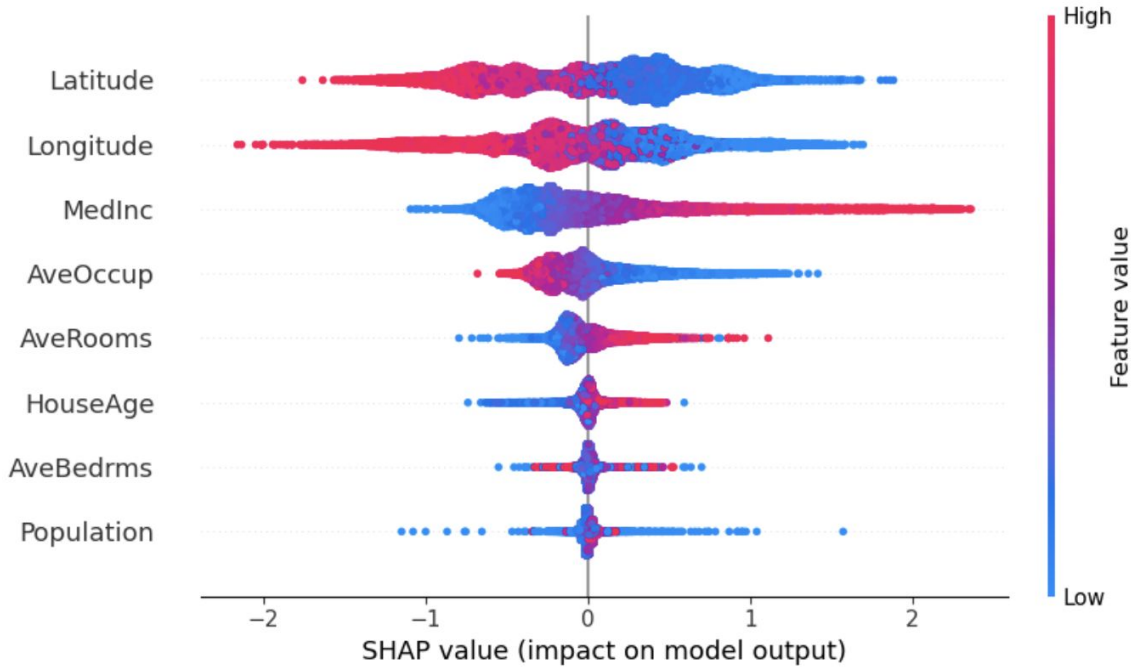
To understand how a single feature effects the output of the model we can plot the SHAP value of that feature vs. the value of the feature for all the examples in a dataset. Since SHAP values represent a feature's responsibility for a change in the model output, the plot below represents the change in predicted house price as the latitude changes. Vertical dispersion at a single value of latitude represents interaction effects with other features. To help reveal these interactions we can color by another feature. If we pass the whole explanation tensor to the `color` argument the scatter plot will pick the best feature to color by. In this case it picks longitude.

```
# create a dependence scatter plot to show the effect of a single feature across the whole d
shap.plots.scatter(shap_values[:, "Latitude"], color=shap_values)
```



Beeswarm Plot - Many Features

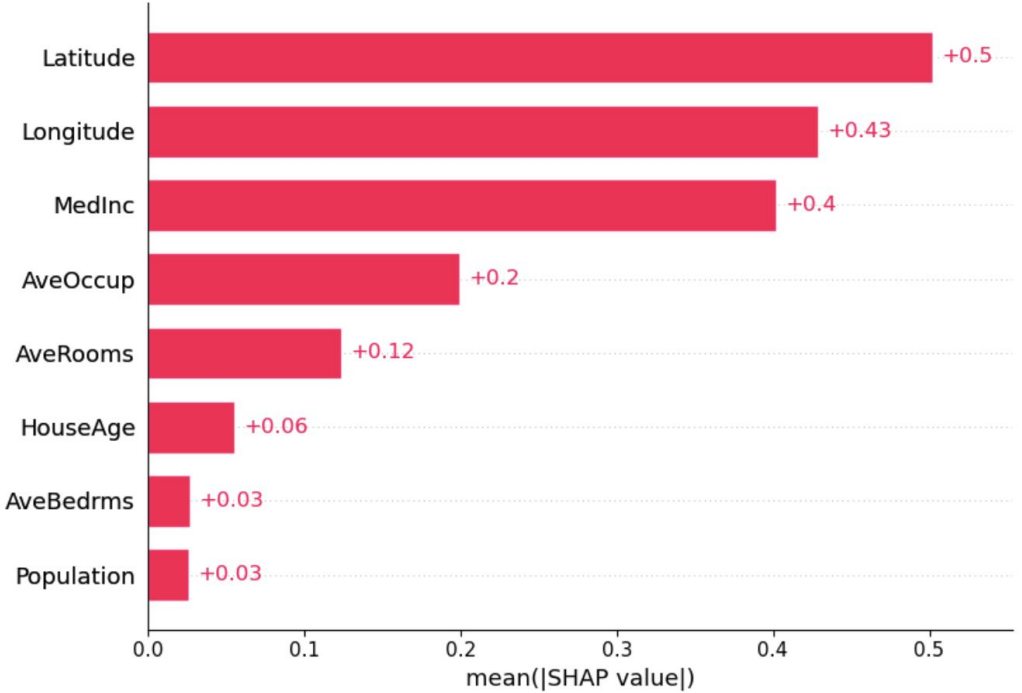
```
# summarize the effects of all the features
shap.plots.beeswarm(shap_values)
```



Bar Plot -Many Features

We can also just take the mean absolute value of the SHAP values for each feature to get a standard bar plot (produces stacked bars for multi-class outputs):

```
shap.plots.bar(shap_values)
```



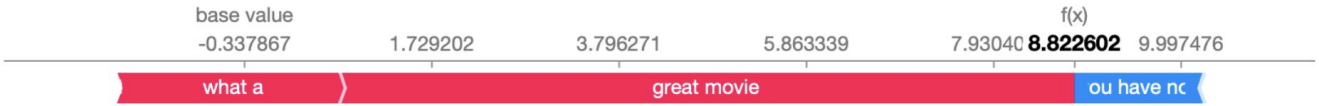
Real life project
Predicting power line failures

For Text

Natural language example (transformers)

SHAP has specific support for natural language models like those in the Hugging Face transformers library. By adding coalitional rules to traditional Shapley values we can form games that explain large modern NLP model using very few function evaluations. Using this functionality is as simple as passing a supported transformers pipeline to SHAP:

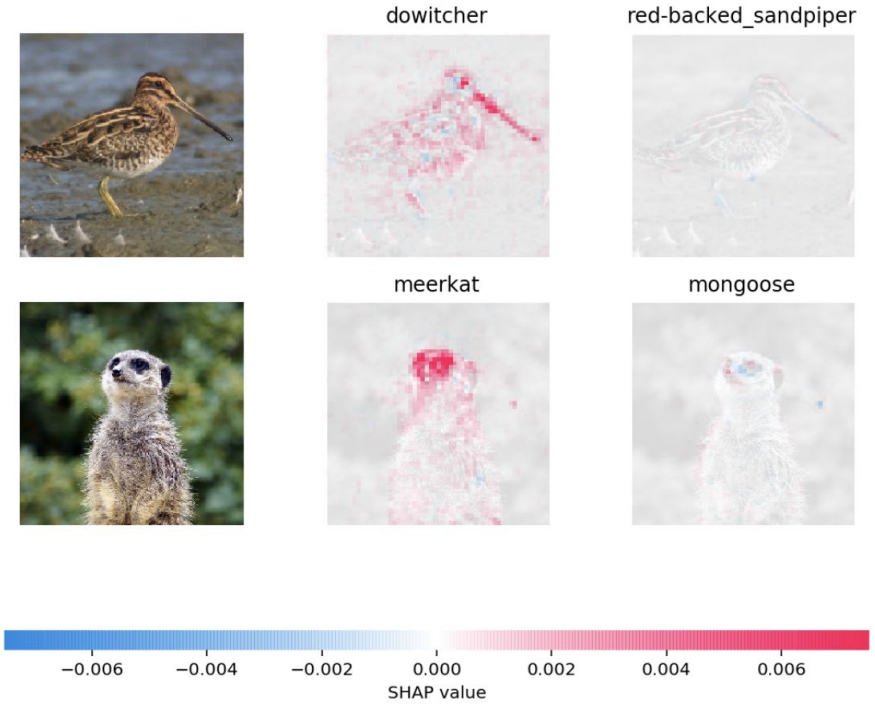
```
import transformers
import shap

# load a transformers pipeline model
model = transformers.pipeline('sentiment-analysis', return_all_scores=True)

# explain the model on two sample inputs
explainer = shap.Explainer(model)
shap_values = explainer(["What a great movie! ...if you have no taste."])

# visualize the first prediction's explanation for the POSITIVE output class
shap.plots.text(shap_values[0, :, "POSITIVE"])
```



what a great movie ! . . . if you have no taste .

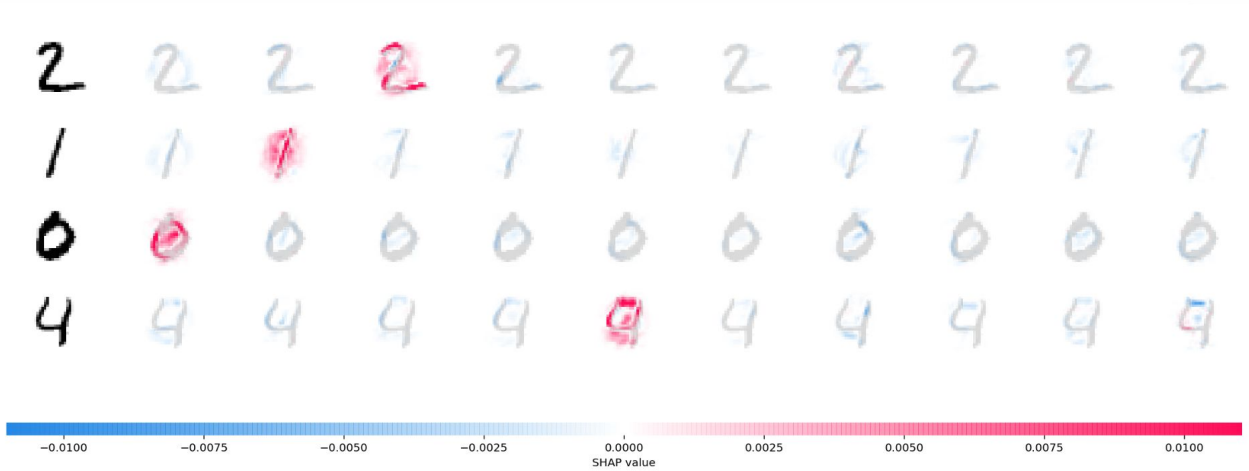
For Images

```
# plot the explanations
shap.image_plot(shap_values, to_explain, index_names)
```



For Images

```
# ...include code from https://github.com/keras-team/keras/blob/master/examples/demo_mnist_c

import shap
import numpy as np

# select a set of background examples to take an expectation over
background = x_train[np.random.choice(x_train.shape[0], 100, replace=False)]

# explain predictions of the model on four images
e = shap.DeepExplainer(model, background)
# ...or pass tensors directly
# e = shap.DeepExplainer((model.layers[0].input, model.layers[-1].output), background)
shap_values = e.shap_values(x_test[1:5])

# plot the feature attributions
shap.image_plot(shap_values, -x_test[1:5])
```



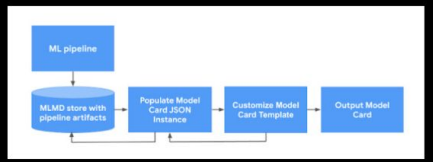
Questions?

Model Card - For Your Model - Example

Google Research Who we are Research areas Our work Programs & events Careers Blog

Home > Blog >

Introducing the Model Card Toolkit for Easier Model Transparency Reporting



July 29, 2020 · Posted by Huanming Fang and Hui Miao, Software Engineers, Google Research

Explainable AI is a team effort

Learn More

Search terms

Interpretable Machine Learning

Interpretable AI

Explainable AI

XAI

Learn More

Technical Books

Molnar, Christoph. (2022). **Interpretable Machine Learning: A Guide for Making Black Box Models Explainable.** Independently published.

Masís, Serg. (2021). **Interpretable Machine Learning with Python: Learn to build interpretable high-performance models with hands-on real-world examples.** Packt Publishing.

Thampi, Ajay. (2022). **Interpretable AI, building explainable machine learning systems**. Manning

Learn More

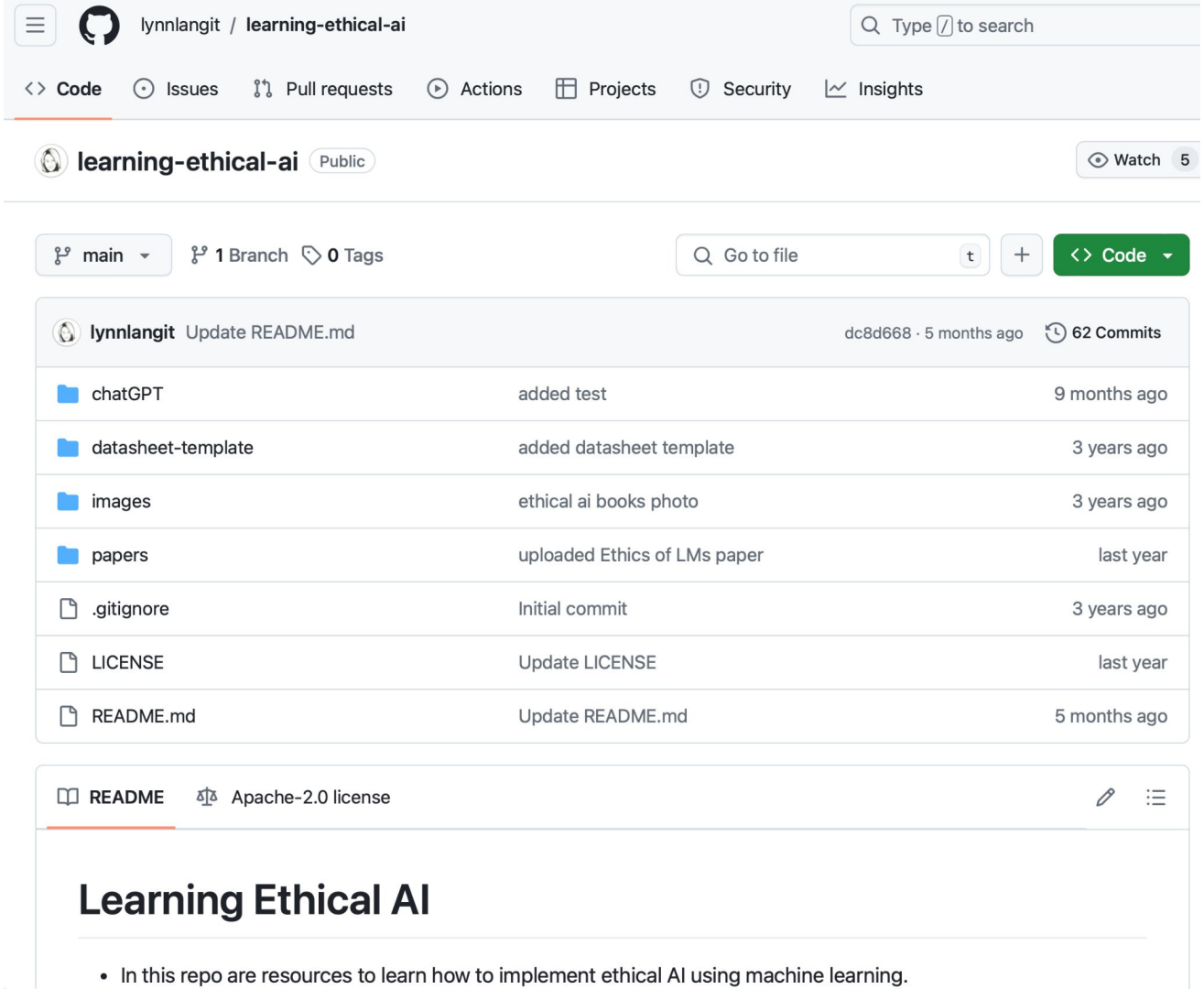
Ethics Books

O’Neil, C. (2016). **Weapons of Math Destruction, How Big Data Increases Inequality and Threatens Democracy.** New York, NY: Broadway Books.

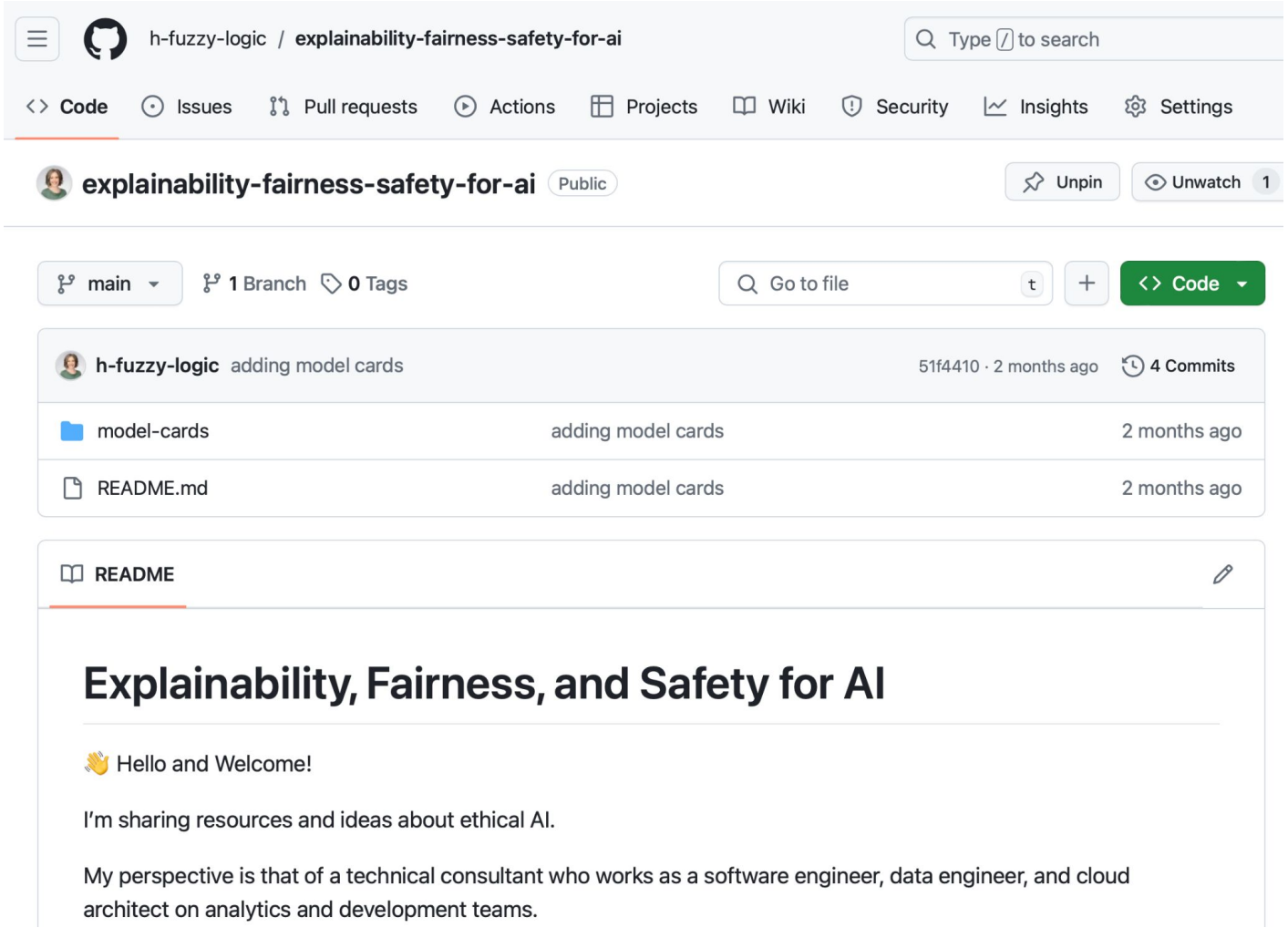
Fry, H. (2018). **Hello world: being human in the age of algorithms** (First edition.). W.W. Norton & Company.

Schellmann, H. (2024). **The algorithm: how AI decides who gets hired, monitored, promoted, and fired and why we need to fight back now.** Hachette Books.

lynnlangit

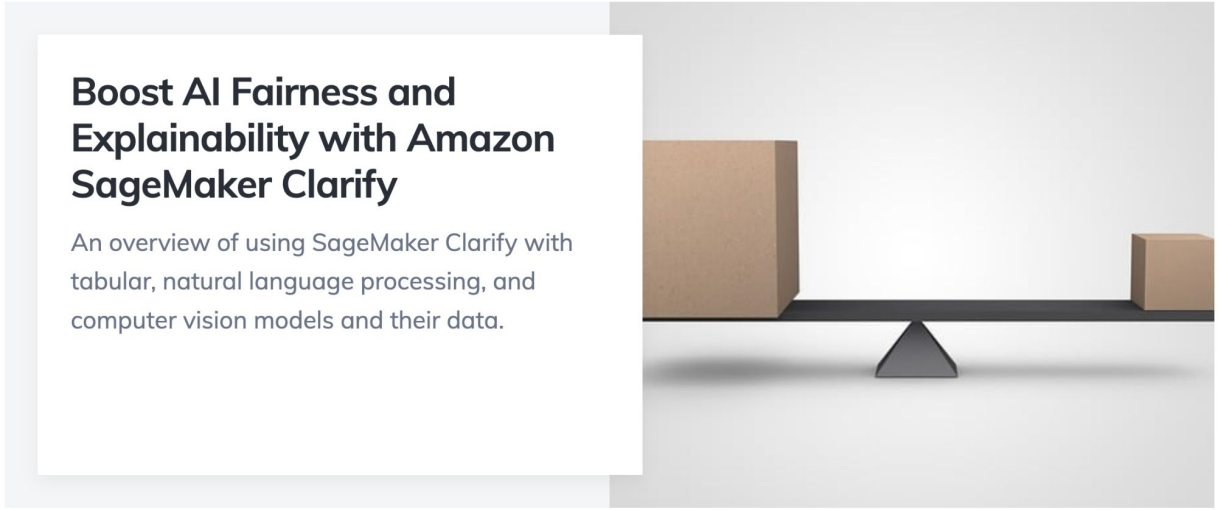

lynnlangit / learning-ethical-ai
Type / to search
Code
Issues
Pull requests
Actions
Projects
Security
Insights
learning-ethical-ai
Public
Watch 5
main
1 Branch
0 Tags
Go to file
t
Code
lynnlangit Update README.md
dc8d668 · 5 months ago
62 Commits
chatGPT
added test
9 months ago
datasheet-template
added datasheet template
3 years ago
images
ethical ai books photo
3 years ago
papers
uploaded Ethics of LMs paper
last year
.gitignore
Initial commit
3 years ago
LICENSE
Update LICENSE
last year
README.md
Update README.md
5 months ago
README
Apache-2.0 license
Learning Ethical AI
In this repo are resources to learn how to implement ethical AI using machine learning.

h-fuzzy-logic



More Technical

AWS - SageMaker Clarify



https://www.heatherwoods.tech/blog/sagemaker-clarify

Contact

Heather Woods

heather.fuzzylogic@gmail.com

https://www.heatherwoods.tech/

https://github.com/h-fuzzy-logic

https://www.linkedin.com/in/heather-woods/

An overview of model agnostic, local, and post hoc Interpretability Methods

Murat Durmus
(CEO AISOMA)

Method	Description	Pros	Cons
LIME	LIME, or Local Interpretable Model-Agnostic Explanations, is an algorithm that can explain the predictions of any classifier or regressor in a faithful way, by approximating it locally with an interpretable model.	- Fully model-agnostic - Many contributors	- The inclusion of unrealistic data instances - Ambiguity of how to select kernel width - Coverage not always clear
SHAP	SHAP, or SHapley Additive exPlanations, is a game theoretic approach to explain the output of any machine learning model. It connects optimal credit allocation with local explanations using the classic Shapley values from game theory and their related extensions.	-It can be seen how both functions move predictions up & down - Based on solid theory. SHAP values have useful and proven mathematical properties - The TreeExplainer is fast	- The inclusion of unrealistic data instances - Computationally expensive in the general case - Based on solid theory. It can be hard to dive into all the math inside this method
Permutation Importance	Permutation Importance is an intuitive way to assess the impact of a feature on the black-box model performance.	- Simple and intuitive - Available through the eli5 library (Python) - Easy to compute	- Labeled test data to compute the loss are required - Different shuffles may give different results - Greatly influenced by correlated features
Partial Dependency Plot	One of the simplest and most understandable methods is a partial dependency graph. This dependency graph shows us the dependency of a target variable from a particular feature.	- Easy and intuitive - Available in sklearn (Python)	- Assumption of feature independence - Loss of higher order interactions
Anchor	A novel model-agnostic system that explained the behavior of complex models with high-precision rules called anchors, representing local, "sufficient" conditions for predictions.	- Clear coverage - High precision - The anchors are expressive	- The inclusion of unrealistic data instances - The code available in the original repository is still in progress